**Happy Number 找規律**

**平方和 如果回到最原本的數字的話 表示持續循環 因此要多一個list去紀錄有沒有重複**

class Solution:

def isHappy(self, n: int) -> bool:

a = []

while 1:

sun = 0

a.append(n)

for i in list(str(n)):

sun += math.pow(int(i),2)

if int(sun) ==1:

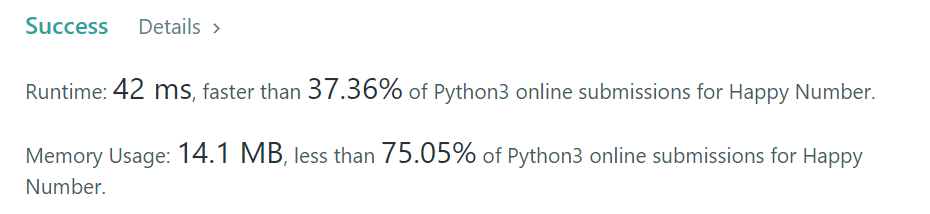
return True

elif int(sun) in a:

return False

else:

n = int(sun)



Others:

def isHappy(self, n: int) -> bool:

def get\_next(n):

total\_sum = 0

while n > 0:

n, digit = divmod(n, 10)

total\_sum += digit \*\* 2

return total\_sum

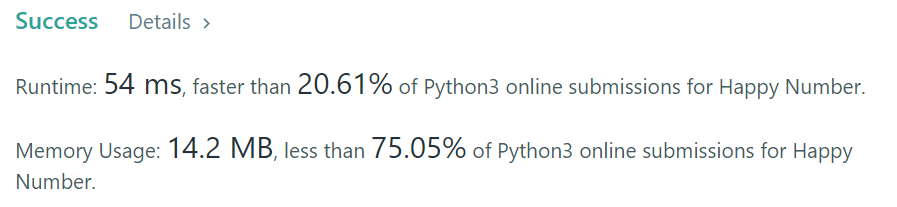
seen = set()

while n != 1 and n not in seen:

seen.add(n)

n = get\_next(n)

return n == 1



class Solution {

private int getNext(int n) {

int totalSum = 0;

while (n > 0) {

int d = n % 10;

n = n / 10;

totalSum += d \* d;

}

return totalSum;

}

public boolean isHappy(int n) {

Set<Integer> seen = new HashSet<>();

while (n != 1 && !seen.contains(n)) {

seen.add(n);

n = getNext(n);

}

return n == 1;

}

}

int digitSquareSum(int n) {

int sum = 0, tmp;

while (n) {

tmp = n % 10;

sum += tmp \* tmp;

n /= 10;

}

return sum;

}

bool isHappy(int n) {

int slow, fast;

slow = fast = n;

do {

slow = digitSquareSum(slow);

fast = digitSquareSum(fast);

fast = digitSquareSum(fast);

if(fast == 1) return 1;

} while(slow != fast);

return 0;

}

def isHappy(self, n):

mem = set()

while n != 1:

n = sum([int(i) \*\* 2 for i in str(n)])

if n in mem:

return False

else:

mem.add(n)

else:

return True